ENCAPSULATING PROTOCOL FOR SESSION PERSISTENCE AND RELIABILITY

Technical Field

[0001] The invention generally relates to network communications. More particularly, the invention relates to a communication protocol that encapsulates other protocols and thereby provides session persistence and reliability.

Background Information

[0002] Communications over a network between two computers, for example a client and a server, can be implemented using a variety of known communication protocols. Often, however, the network connection is susceptible to breakdown. A wireless connection between a client and a server, for example, is often unreliable. In other cases, the network connection is intermittent. For example, a connection can be lost when one enters an elevator or tunnel and may only be capable of being restored following one's exit from the elevator or tunnel.

[0003] When communicating over a network connection using many current protocols, data packets are lost when the network connection is disrupted. For example, when many current protocols are communicated over a standard TCP network connection, data buffers are typically flushed upon disruption of the connection. As such, when the network connection is restored, a networked application, for example, is unable to resume from where it was prior to disruption. Typically an error message is displayed, adding user frustration to inconvenience.

[0004] Moreover, communicating over a network with many current protocols often requires frequent tear down and re-establishment of the transport connection. For example, using HTTP, either on its own or in conjunction with typical proxy protocols, to browse a website over a standard TCP connection requires, in addition to a new HTTP connection for each resource, the

closure of a previous TCP/proxy protocol connection and the opening of a new TCP/proxy protocol connection for each resource.

[0005] Furthermore, when a network connection fails, one must typically restart and completely re-logon to the server before communications can resume. For example, logon credentials need to be re-applied. Often, this is a slow process that also results in user inefficiency.

[0006] Improved systems and methods for network communications are, therefore, needed.

Summary of the Invention

[0007] The present invention relates to systems and methods for providing a client with a reliable connection to a host service. A first communication protocol, capable of encapsulating secondary protocols used in communications between the client and the host service, ensures that data is maintained during a disrupted network connection. More specifically, data communicated between the client and the host service is buffered. When, for example, a client, such as a mobile client, roams between different access points in the same network, the buffered data is maintained during the temporarily disrupted network connection. Similarly, in another example, when a client switches between networks (*e.g.,* from a wired network to a wireless network) the buffered data is maintained during the temporarily disrupted connection to the host service. In addition to maintaining buffered data when a client roams between network access points or between networks themselves, buffered data can also be maintained, for example, when the network connection is disrupted due to a failure of a server side component (*e.g.,* a failure of a server side proxy), due to a time-out in the system, or due to other reasons. Accordingly, session persistence is achieved and reliability ensured.

[0008] Using the first communication protocol of the present invention also allows the secondary protocol connections tunneled therein to be opened and/or closed, repetitively, without also

requiring the transport connection over which the first protocol is communicated, or the first protocol connection itself, to similarly be repetitively opened and/or closed. As such, the efficiency of the system is improved.

[0009] Moreover, the present invention relates to systems and methods for re-connecting a client to a host service following a disruption to a network connection. More particularly, the systems and methods for re-connecting the client to the host service use re-connection tickets and do not require the re-application of user logon credentials. As such, the time needed to re-connect the client to the host service is reduced.

[0010] In one aspect, the invention generally relates to a method for network communications. The method includes establishing a first connection between a client and a first protocol service using a first protocol and communicating between the client and the first protocol service via a plurality of secondary protocols encapsulated within the first protocol. Moreover, at least one of the secondary protocols includes a plurality of virtual channels.

[0011] In one embodiment of this aspect of the invention, a second connection is established between the first protocol service and a host service using one of the secondary protocols. Communication between the first protocol service and the host service occurs via one of the secondary protocols. In another embodiment, a plurality of second connections are established between the first protocol service and a plurality of host services using the plurality of the secondary protocols. Specifically, each of the plurality of second connections is established between the first protocol service and a different host service and each of the plurality of second connections is established using one of the plurality of secondary protocols. Communication between the first protocol service and the plurality of host services occurs over each of the plurality of second connections via one of the plurality of secondary protocols. In yet another

embodiment, the first connection between the client and the first protocol service is established through an intermediary node.

[0012] The first protocol can be communicated over TCP/IP and the secondary protocol can be, for example, HTTP, RDP, ICA, FTP, Oscar, or Telnet. Additionally, each virtual channel can include a plurality of protocol packets that enable remote access functionality.

[0013] In one embodiment, the communications are compressed at the level of the first protocol. In another embodiment, the communications are encrypted at the level of the first protocol. In yet another embodiment, the first connection is secure, a second connection between the first protocol service and a first host service is established, the client and the first host service communicate via the first connection and the second connection, the second connection is broken, a third connection between the first protocol service and a second host service is established without interrupting the first connection, and the client and the second host service communicate via the first connection and the third connection.

[0014] In another aspect, the invention relates to a method for providing a client with a reliable connection to a host service. The method includes establishing a first connection between the client and a first protocol service using a first protocol and establishing a second connection between the first protocol service and the host service using a secondary protocol. The first protocol is for encapsulating a plurality of secondary protocols. The method further includes maintaining a queue of data packets most recently transmitted via the first connection on at least one of the client and the first protocol service. Upon failure of the first connection: the second connection is maintained, the queue of data packets most recently transmitted via the first connection is still maintained, and a third connection is established between the client and the first protocol service using the first protocol.

[0015] In one embodiment of this aspect of the invention, at least one of the queued data packets is transmitted via the third connection.

[0016] In another aspect, the invention provides a method for re-connecting a client to a host service. The method includes providing a first connection between the client and an intermediary node, a second connection between the intermediary node and a first protocol service, and a third connection between the first protocol service and the host service. A disruption is detected in at least one of the first connection and the second connection. The first connection between the client and the intermediary node is re-established while the third connection between the first protocol service and the host service is maintained. A first ticket and a second ticket are also received at the intermediary node. The first ticket is validated. After the first ticket is validated, the second connection between the intermediary node and the first protocol service is re-established. The second ticket is validated and, after the second ticket is validated, the re-established second connection is linked to the maintained third connection.

[0017] In one embodiment of this aspect of the invention, the method includes interrupting, after the disruption in at least one of the first connection and the second connection is detected, any remaining connections of the first connection and the second connection.

[0018] In another embodiment, the first ticket is transmitted from the intermediary node to a ticket authority and the first ticket is validated using the ticket authority. After the first ticket is validated, an address for the first protocol service is received at the intermediary node. Moreover, the first ticket can be deleted after it is validated. After the first ticket is deleted, a replacement first ticket can be generated.

[0019] In yet another embodiment, the second ticket is transmitted from the intermediary node to the first protocol service and the second ticket is validated using the first protocol service. The

second ticket can be deleted after it is validated. After the second ticket is deleted, a replacement second ticket can be generated.

[0020] In still another embodiment, the intermediary node can transmit to the ticket authority a request for the first ticket. The first ticket, which can be, for example, a random number, can be generated at the ticket authority. The ticket authority can also generate a handle and save, at the ticket authority, a copy of the first ticket, a copy of the handle, and an address for the first protocol service. The first ticket and the handle can be transmitted from the ticket authority to the intermediary node, which can then transmit the first ticket to the client. The handle can also be used to delete the copy of the first ticket saved at the ticket authority.

[0021] In a further embodiment, the second ticket, which can be, for example, a random number, can be generated at the first protocol service. A copy of the second ticket and a session number can also be saved at the at the first protocol service. The second ticket can be transmitted from the first protocol service to the client. Additionally, at least one of the first ticket and the second ticket can be automatically deleted after a pre-determined period of time.

[0022] In another aspect, the invention provides a method for re-connecting a client to a host service. The method includes providing a first connection between the client and a first intermediary node, a second connection between the first intermediary node and a first protocol service, and a third connection between the first protocol service and the host service. A disruption is detected in at least one of the first connection and the second connection. A fourth connection between the client and a second intermediary node, which is different from the first intermediary node, is established while the third connection between the first protocol service and the host service is maintained. A first ticket and a second ticket are also received at the second intermediary node. The first ticket is validated. After the first ticket is validated, a fifth

connection between the second intermediary node and the first protocol service is established. The second ticket is validated and, after the second ticket is validated, the established fifth connection is linked to the maintained third connection.

[0023] In another aspect, the invention provides a method for re-connecting a client to a host service. The method includes providing a first connection between the client and a first protocol service, and a second connection between the first protocol service and the host service. A disruption is detected in the first connection. The first connection between the client and the first protocol service is re-established while the second connection between the first protocol service and the host service is maintained. A ticket is also received at the first protocol service. The ticket is validated. After the ticket is validated, the re-established first connection is linked to the maintained second connection.

[0024] In one embodiment of this aspect of the invention, the ticket, after it is validated, is deleted. Moreover, after the ticket is deleted, a replacement ticket can be generated. In another embodiment, the ticket, which can be a random number, is generated at the first protocol service. A copy of the ticket and a session number can be saved at the first protocol service. The ticket can also be transmitted from the first protocol service to the client. Additionally, the ticket can be automatically deleted after a pre-determined period of time.

[0025] In another aspect, the invention generally relates to a system for network communications. The system includes a first protocol service configured to accept a first connection with a client and communicate with the client via a plurality of secondary protocols encapsulated within a first protocol. Moreover, at least one of the secondary protocols includes a plurality of virtual channels.

[0026] In one embodiment of this aspect of the invention, the first protocol service is further configured to establish a second connection with a host service and communicate with the host service via one of the secondary protocols. In another embodiment, the first protocol service is further configured to establish a plurality of second connections with a plurality of host services using the plurality of secondary protocols. Specifically, each of the plurality of second connections is established with a different host service and each of the plurality of second connections is established using one of the plurality of secondary protocols. In such an embodiment, the first protocol service is further configured to communicate with the plurality of host services over each of the plurality of second connections via one of the plurality of secondary protocols. In yet another embodiment, the first connection with the client is routed through an intermediary node.

[0027] The first protocol can be communicated over TCP/IP and the secondary protocol can be, for example, HTTP, RDP, ICA, FTP, Oscar, or Telnet. Additionally, each virtual channel can include a plurality of protocol packets that enable remote access functionality.

[0028] In one embodiment, the first protocol service is configured to compress the communications at the level of the first protocol. In another embodiment, the first protocol service is configured to encrypt the communications at the level of the first protocol. In yet another embodiment, the first connection is secure, and the first protocol service is configured to establish a second connection with a first host service, interrupt the second connection, and establish a third connection with a second host service without interrupting the first connection.

[0029] In another aspect, the invention relates to a system for providing a client with a reliable connection to a host service. The system includes a first protocol service and the host service. The first protocol service is configured to accept a first connection with the client, establish a

second connection with the host service, and, upon failure of the first connection,: maintain the second connection and accept a third connection from the client. The host service is configured to accept the second connection with the first protocol service and, upon failure of the first connection, maintain the second connection. The first connection and the third connection are each established using a first protocol, which can encapsulate a plurality of secondary protocols. Moreover, at least one of the client and the first protocol service is further configured to maintain, before and upon failure of the first connection, a queue of data packets most recently transmitted via the first connection.

[0030] In one embodiment of this aspect of the invention, the client is further configured to transmit at least one of the queued data packets via the third connection. Alternatively, the first protocol service can be configured to transmit at least one of the queued data packets via the third connection.

[0031] In another aspect, the invention provides a system for re-connecting a client to a host service. The system includes the client, an intermediary node, and a first protocol service. The client is configured to maintain a first connection with the intermediary node. For its part, the intermediary node is configured to maintain the first connection with the client and a second connection with the first protocol service. The first protocol service is configured to maintain the second connection with the intermediary node and a third connection with the host service. In accordance with this system, a disruption is detected in at least one of the first connection and the second connection, the first connection is re-established between the client and the intermediary node while the third connection between the first protocol service and the host service is maintained, a first ticket and a second ticket are transmitted from the client to the intermediary node, the first ticket is validated, the second connection between the intermediary node and the

first protocol service is re-established after the first ticket is validated, the second ticket is validated, and, after the second ticket is validated, the re-established second connection is linked to the maintained third connection.

[0032] In one embodiment of this aspect of the invention, after the disruption in at least one of the first connection and the second connection is detected, any remaining connections of the first connection and the second connection are broken.

[0033] In another embodiment, the first ticket is validated using a ticket authority. The ticket authority is, for example, configured to receive the first ticket from the intermediary node and validate the first ticket. In one embodiment, the intermediary node is further configured to receive, after the first ticket is validated, an address for the first protocol service. The ticket authority can be configured to delete the first ticket after it is validated. Moreover, the ticket authority can be configured to generate, after the first ticket is deleted, a replacement first ticket.

[0034] In another embodiment, the second ticket is validated using the first protocol service. The first protocol service is, for example, configured to receive the second ticket from the intermediary node and validate the second ticket. The first protocol service can be configured to delete the second ticket after it is validated. Moreover, the first protocol service can be configured to generate, after the second ticket is deleted, a replacement second ticket.

[0035] In still another embodiment, the intermediary node is configured to transmit a request for the first ticket to the ticket authority. The ticket authority can be configured to generate the first ticket, which can be, for example, a random number. The ticket authority can also be configured to generate a handle and to save a copy of the first ticket, a copy of the handle, and an address for the first protocol service. The ticket authority can be configured to transmit the first ticket and the handle to the intermediary node, which can be configured to then transmit the first ticket to

the client. The intermediary node can also be configured to use the handle to delete the copy of the first ticket saved at the ticket authority.

[0036] In a further embodiment, the first protocol service can be configured to generate the second ticket, which can be, for example, a random number. The first protocol service can also be configured to save a copy of the second ticket and a session number. In another embodiment, the first protocol service is configured to transmit the second ticket to the client. Additionally, at least one of the first ticket and the second ticket can be configured for automatic deletion after a pre-determined period of time.

[0037] In another aspect, the invention provides a system for re-connecting a client to a host service. The system includes the client, a first intermediary node, a first protocol service, and a second intermediary node, which is different from the first intermediary node. The client is configured to maintain a first connection with the first intermediary node. For its part, the first intermediary node is configured to maintain the first connection with the client and a second connection with the first protocol service. The first protocol service is configured to maintain the second connection with the first intermediary node and a third connection with the host service. In accordance with this system, a disruption is detected in at least one of the first connection and the second connection, a fourth connection is established between the client and a second intermediary node while the third connection between the first protocol service and the host service is maintained, a first ticket and a second ticket are transmitted from the client to the second intermediary node, the first ticket is validated, a fifth connection between the second intermediary node and the first protocol service is established after the first ticket is validated, the second ticket is validated, and, after the second ticket is validated, the established fifth connection is linked to the maintained third connection.

[0038] In another aspect, the invention provides a system for re-connecting a client to a host service. The system includes the client and a first protocol service. The client is configured to maintain a first connection with the first protocol service. For its part, the first protocol service is configured to maintain the first connection with the client and a second connection with the host service. In accordance with this system, a disruption is detected in the first connection, the first connection is re-established between the client and the first protocol service while the second connection between the first protocol service and the host service is maintained, a ticket is transmitted from the client to the first protocol service, the ticket is validated, and, after the ticket is validated, the re-established first connection is linked to the maintained second connection.

[0039] In one embodiment of this aspect of the invention, the first protocol service is further configured to delete, after the ticket is validated, the ticket. Moreover, the first protocol service can be further configured to generate, after the ticket is deleted, a replacement ticket. In another embodiment, the first protocol service is further configured to generate the ticket, which can be, for example, a random number. The first protocol service can be configured to save a copy of the ticket and a session number. The first protocol service can also be configured to transmit the ticket to the client. Additionally, the ticket can be configured for automatic deletion after a pre-determined period of time.

Brief Description of the Drawings

[0040] The foregoing and other objects, aspects, features, and advantages of the invention will become more apparent and may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

[0041] FIG. 1A is a block diagram of a system for providing a client with a reliable connection to a host service according to an illustrative embodiment of the invention;

[0042] FIG. 1B is a block diagram of a system for providing a client with a reliable connection to a host service according to another illustrative embodiment of the invention;

[0043] FIG. 2 depicts communications occurring over a network according to an illustrative embodiment of the invention;

[0044] FIG. 3 depicts communications occurring over a network according to another illustrative embodiment of the invention;

[0045] FIG. 4 depicts a process for encapsulating a plurality of secondary protocols within a first protocol for communication over a network according to an illustrative embodiment of the invention;

[0046] FIG. 5 is a block diagram of the illustrative system of FIG. 1A further including components for re-connecting the client to a host service according to an illustrative embodiment of the invention;

[0047] FIG. 6A is a block diagram of the illustrative system of FIG. 5 further including components for initially connecting the client to a host service according to an illustrative embodiment of the invention;

[0048] FIG. 6B is a block diagram of the illustrative system of FIG. 6A further including a component for initially connecting the client to the host service and for re-connecting the client to the host service according to an illustrative embodiment of the invention;

[0049] FIG. 6C is a block diagram of an alternative embodiment of the system of FIG. 6B;

[0050] FIG. 7 is a flow diagram of a method for network communications according to an illustrative embodiment of the invention;

[0051] FIGS. 8A-8C are flow diagrams of a method for connecting a client to a plurality of host services according to an illustrative embodiment of the invention;

[0052] FIG. 9 is a flow diagram of a method for providing a client with a reliable connection to host services and for re-connecting the client to the host services according to an illustrative embodiment of the invention; and

[0053] FIGS. 10A-10B are flow diagrams of a method for re-connecting a client to host services according to an illustrative embodiment of the invention.

Description

[0054] Certain embodiments of the present invention are described below. It is, however, expressly noted that the present invention is not limited to these embodiments, but rather the intention is that additions and modifications to what is expressly described herein also are included within the scope of the invention. Moreover, it is to be understood that the features of the various embodiments described herein are not mutually exclusive and can exist in various combinations and permutations, even if such combinations or permutations are not made express herein, without departing from the spirit and scope of the invention.

[0055] Referring to FIG. 1A, in general, the invention pertains to network communications and can be particularly useful for providing a client with a reliable connection to a host service. In broad overview, a system 100 for network communications includes a remote client 108 (*e.g.*, a first computing device) in communication with a first protocol service 112 (*e.g.*, a second computing device) over a network 104. Also included in the system 100 are a plurality of host services 116a-116n (*e.g.*, third computing devices) that are in communication, over a network 104', with the first protocol service 112 and, through the first protocol service 112 and over the network 104, with the client 108. Alternatively, in another illustrative embodiment of the

invention, and with reference now to FIG. 1B, the first protocol service 112 and the host services 116a-116n are not implemented as separate computing devices, as in FIG. 1A, but, rather, they are incorporated into the same computing device, such as, for example, host node 118a. The system 100 can include one, two, or any number of host nodes 118a-118n.

[0056] In one embodiment, the networks 104 and 104′ are separate networks, as in FIG. 1A. The networks 104 and 104′ can be the same network 104, as in FIG. 1B. In one embodiment, the network 104 and/or the network 104′ is, for example, a local-area network (LAN), such as a company Intranet, or a wide area network (WAN), such as the Internet or the World Wide Web. The remote client 108, the first protocol service 112, the host services 116, and/or the host nodes 118 can be connected to the networks 104 and/or 104' through a variety of connections including, but not limited to, standard telephone lines, LAN or WAN links (*e.g.,* 802.11, T1, T3, 56kb, X.25), broadband connections (*e.g.,* ISDN, Frame Relay, ATM), wireless connections, or some combination of any or all of the above.

[0057] Moreover, the client 108 can be any workstation, desktop computer, laptop, handheld computer, mobile telephone, or other form of computing or telecommunications device that is capable of communication and that has sufficient processor power and memory capacity to perform the operations described herein. The client 108 can include, for example, a visual display device (*e.g.,* a computer monitor), a data entry device (*e.g.,* a keyboard), persistent and/or volatile storage (*e.g.,* computer memory), a processor, and a mouse.

[0058] Similarly, with reference to FIG. 1A, each of the first protocol service 112 and the host services 116 can be provided on any computing device that is capable of communication and that has sufficient processor power and memory capacity to perform the operations described herein. Alternatively, where the functionality of the first protocol service 112 and the host services 116

are incorporated into the same computing device, such as, for example, a host node 118, as in FIG. 1B, the first protocol service 112 and/or the host services 116 can be implemented as a software program running on a general purpose computer and/or as a special purpose hardware device, such as, for example, an ASIC or an FPGA, and the host node 118 can be any computing device that is capable of communication and that has sufficient processor power and memory capacity to perform the operations described herein.

[0059] In one embodiment, each of the host services 116 hosts one or more application programs that are remotely available to the client 108. The same application program can be hosted by one or any number of the host services 116. Examples of such applications include word processing programs, such as MICROSOFT WORD, and spreadsheet programs, such as MICROSOFT EXCEL, both of which are available from Microsoft Corporation of Redmond, Washington. Other examples of application programs that may be hosted by any/all of the host services 116 include financial reporting programs, customer registration programs, programs providing technical support information, customer database applications, and application set managers. Moreover, in one embodiment, the host services 116 are audio/video streaming servers that provide streaming audio and/or streaming video to the client 108. In another embodiment, the host services 116 include file servers that provide any/all file types to the client 108.

[0060] Referring still to the illustrative embodiments of FIGS. 1A and 1B, the client 108 is configured to establish a connection 120 between the client 108 and a first protocol service 112 over the network 104 using a first protocol. For its part, the first protocol service 112 is configured to accept the connection 120. The client 108 and the first protocol service 112 can, therefore, communicate with one another using the first protocol.

[0061] In some embodiments, as shown in FIGS. 1A and 1B, a client agent 128 is included within the client 108. The client agent 128 can be, for example, implemented as a software program and/or as a hardware device, such as, for example, an ASIC or an FPGA. The client agent 128 can use any type of protocol and it can be, for example, an HTTP client agent, an FTP client agent, an Oscar client agent, a Telnet client agent, an Independent Computing Architecture (ICA) client agent from Citrix Systems, Inc. of Fort Lauderdale, Florida, or a Remote Desktop Procedure (RDP) client agent from Microsoft Corporation of Redmond, Washington. In some embodiments, the client agent 128 is itself configured to communicate using the first protocol. In some embodiments (not shown), the client 108 includes a plurality of client agents 128a-128n, each of which communicates with a host service 116a-116n, respectively.

[0062] In another embodiment, a standalone client agent is configured to enable the client 108 to communicate using the first protocol. The standalone client agent can be incorporated within the client 108 or, alternatively, the standalone client agent can be separate from the client 108. The standalone client agent is, for example, a local host proxy. In general, the standalone client agent can implement any of the functions described herein with respect to the client agent 128.

[0063] As also described further below, the first protocol service 112 is, in one embodiment, itself configured to communicate using the first protocol.

[0064] The first protocol service 112 is configured to establish a connection 124a-124n between the first protocol service 112 and the host service 116a-116n, respectively. For example, the first protocol service 112 can establish a connection 124a between the first protocol service 112 and one host service 116a and a connection 124b between the first protocol service 112 and another host service 116b. In one embodiment, the first protocol service 108 separately establishes such connections 124a-124n (*i.e.*, the first protocol service 112 establishes one connection at a time).

In another embodiment, the first protocol service 112 simultaneously establishes two or more of such connections 124a-124n.

[0065] In yet another embodiment, the first protocol service 112 is configured to provide two or more connections 124 without interrupting the connection 120 with the client 108. For example, the first protocol service 112 can be configured to establish the connection 124a between the first protocol service 112 and the host service 116a when a user of the client 108 requests execution of a first application program residing on the host service 116a. When the user ends execution of the first application program and initiates execution of a second application program residing, for example, on the host service 116b, the first protocol service 112 is, in one embodiment, configured to interrupt the connection 124a and establish the connection 124b between the first protocol service 112 and the host service 116b, without disrupting the connection 120 between the first protocol service 112 and the client 108.

[0066] The first protocol service 112 and the host services 116a-116n can communicate over the connections 124a-124n, respectively, using any one of a variety of secondary protocols, including, but not limited to, HTTP, FTP, Oscar, Telnet, the ICA presentation protocol from Citrix Systems, Inc. of Fort Lauderdale, Florida, and/or the RDP presentation protocol from Microsoft Corporation of Redmond, Washington. For example, the first protocol service 112 and the host service 116a can communicate over the connection 124a using the ICA presentation protocol, while the first protocol service 112 and the host service 116b can communicate over the connection 124b using the RDP presentation protocol.

[0067] In one embodiment, the secondary protocol used for communicating between the first protocol service 112 and a host service 116, such as, for example, the ICA presentation protocol, includes a plurality of virtual channels. A virtual channel is a session-oriented transmission

connection that is used by application-layer code to issue commands for exchanging data. For example, each of the plurality of virtual channels can include a plurality of protocol packets that enable functionality at the remote client 108. In one embodiment, one of the plurality of virtual channels includes protocol packets for transmitting graphical screen commands from a host service 116, through the first protocol service 112, to the client 108, for causing the client 108 to display a graphical user interface. In another embodiment, one of the plurality of virtual channels includes protocol packets for transmitting printer commands from a host service 116, through the first protocol service 112, to the client 108, for causing a document to be printed at the client 108.

[0068] In one embodiment, the first protocol is a tunneling protocol. The first protocol service 112 encapsulates a plurality of secondary protocols, each used for communication between one of the host services 116 and the first protocol service 112, within the first protocol. As such, the host services 116 and the first protocol service 112 communicate with the client 108 via the plurality of secondary protocols. In one embodiment, the first protocol is, for example, an application-level transport protocol, capable of tunneling the multiple secondary protocols over a TCP/IP connection.

[0069] Referring to FIG. 2, conceptually, communications between the client 108 and the first protocol service 112 via the connection 120 take the form of a plurality of secondary protocols 200a-200n (*e.g.*, HTTP, FTP, Oscar, Telnet, ICA, and/or RDP) encapsulated within a first protocol 204. This is indicated by the location of secondary protocols 200a-200n inside the first protocol 204. Where secure communication is not called for, the first protocol 204 can be, as illustrated in FIG. 2, communicated over a TCP connection 208.

[0070] Referring now to FIG. 3, if secure communication is used, the first protocol 204 is communicated over an encrypted connection, such as, for example, a TCP connection 212 secured by using the Secure Socket Layer (SSL) 216 protocol. SSL is a secure protocol first developed by Netscape Communication Corporation of Mountain View, California, and is now a standard promulgated by the Internet Engineering Task Force (IETF) as the Transport Layer Security (TLS) protocol and described in IETF RFC-2246.

[0071] Thus, the plurality of secondary protocols 200a-200n are communicated within the first protocol 204 with (FIG. 3) or without (FIG. 2) security over the connection 120.

[0072] In one embodiment, the first protocol 204 allows the secondary protocol connections 200 tunneled therein, such as, for example, an HTTP connection 200, to be opened and/or closed, repetitively, without also requiring the transport connection over which the first protocol 204 is communicated (*e.g.,* TCP connection 208 and/or 212), the SSL protocol connection 216, or the first protocol connection 204 itself to similarly be repetitively opened and/or closed.

[0073] Referring to FIG. 4, an example process 300 used by the first protocol service 112 and the client agent 128 of the client 108 encapsulates the plurality of secondary protocols 200 (*e.g.,* HTTP, FTP, Oscar, Telnet, ICA, and/or RDP) within the first protocol 204 for communication via the connection 120. Optionally, as described below, the example process 300 used by the first protocol service 112 and the client agent 128 of the client 108 also compresses and/or encrypts the communications at the level of the first protocol prior to communications via the connection 120. From the point of view of the first protocol service 112, secondary protocol packets 304 are received via the connections 124 at the first protocol service 112. For example, two secondary protocol packets 304a and 304b are received by the first protocol service 112. One, two, or any number of secondary protocol packets 304 can be received. In one

embodiment, the secondary protocol packets 304 are transmitted by the host services 116 to the first protocol service 112 over the connection 124. The secondary protocol packets 304 include a header 308 and a data payload 312.

[0074] Following receipt of the secondary protocol packets 304, the first protocol service 112 encapsulates one or more of the secondary protocol packets 304 within a first protocol packet 316. In one embodiment, the first protocol service 112 generates a first protocol packet header 320 and encapsulates within the data payload 324 of the first protocol packet 316 one or more secondary protocol packets 304, such as, for example, two secondary protocol packets 304a and 304b. In another embodiment, only one secondary protocol packet 304a is encapsulated in each first protocol packet 316.

[0075] In one embodiment, the first protocol packets 316 are then transmitted over the connection 120, for example over the connection 208 described with reference to FIG. 2, to the client agent 128 of the client 108. Alternatively, in another embodiment, the first protocol service 112 is further configured to encrypt, prior to the transmission of any first protocol packets 316, communications at the level of the first protocol 204. In one such embodiment, the first protocol packets 316 are encrypted by using, for example, the SSL protocol described with reference to FIG. 3. As a result, a secure packet 328, including a header 332 and an encrypted first protocol packet 316' as a data payload 336, is generated. The secure packet 328 can then be transmitted over the connection 120, for example over the secure TCP connection 212 illustrated in FIG. 3, to the client agent 128 of the client 108.

[0076] In another embodiment, the first protocol service 112 is further configured to compress, prior to the transmission of any first protocol packets 316, communications at the level of the first protocol 204. In one embodiment, prior to encrypting the first protocol packet 316, the first

protocol service 112 compresses, using a standard compression technique, the first protocol packet 316. As such, the efficiency of the system 100 is improved.

[0077] Referring again to FIGS. 1A-1B, the system 100 of the present invention, in one embodiment, provides the remote client 108 with a reliable connection to a host service 116, such as, for example, the host service 116a. For example, if the client 108 establishes a connection 120 between the client 108 and the first protocol service 112 and the first protocol service 112 establishes a connection 124a between the first protocol service 112 and the host service 116a, then either the client agent 128, the first protocol service 112, or both are configured to maintain a queue of the first protocol data packets most recently transmitted via the connection 120. For example, the queued data packets can be maintained by the client agent 128 and/or the first protocol service 112 both before and upon a failure of the connection 120. Moreover, upon a failure of the connection 120, the first protocol service 112 and, likewise, the host service 116a are configured to maintain the connection 124a.

[0078] Following a failure of the connection 120, the client 108 establishes a new connection 120 with the first protocol service 112, without losing any data. More specifically, because the connection 124a is maintained upon a failure of the connection 120, a newly established connection 120 can be linked to the maintained connection 124a. Further, because the most recently transmitted first protocol data packets are queued, they can be again transmitted by the client 108 to the first protocol service 112 and/or by the first protocol service 112 to the client 108 over the newly established connection 120. As such, the communication session between the host service 116a and the client 108, through the first protocol service 112, is persistent and proceeds without any loss of data.

[0079] In one embodiment, the client agent 128 of the client 108 and/or the first protocol service 112 number the data packets that they transmit over the connection 120. For example, each of the client agent 12_o and the first protocol service 112 separately numbers its own transmitted data packets, without regard to how the other is numbering its data packets. Moreover, the numbering of the data packets can be absolute, without any re-numbering of the data packets, *i.e.*, the first data packet transmitted by the client agent 128 and/or the first protocol service 112 can be numbered as No. 1, with each data packet transmitted over the connection 120 by the client agent 128 and/or the first protocol service 112, respectively, consecutively numbered thereafter.

[0080] In one such embodiment, following a disrupted and re-established connection 120, the client agent 128 and/or the first protocol service 112 informs the other of the next data packet that it requires. For example, where the client agent 128 had received data packets Nos. 1-10 prior to the disruption of connection 120, the client agent 128, upon re-establishment of the connection 120, informs the first protocol service 112 that it now requires data packet No. 11. Similarly, the first protocol service 112 can also operate as such. Alternatively, in another such embodiment, the client agent 128 and/or the first protocol service 112 informs the other of the last data packet received. For example, where the client agent 128 had received data packets Nos. 1-10 prior to the disruption of connection 120, the client agent 128, upon re-establishment of the connection 120, informs the first protocol service 112 that it last received data packet No. 10. Again, the first protocol service 112 can also operate as such. In yet another embodiment, the client agent 128 and/or the first protocol service 112 informs the other, upon re-establishment of the connection 120, of both the last data packet received and the next data packet it requires.

[0081] In such embodiments, upon re-establishment of the connection 120, the client agent 128 and/or the first protocol service 112 can re-transmit the buffered data packets not received by the other, allowing the communication session between a host service 116 and the client 108, through the first protocol service 112, to proceed without any loss of data. Moreover, upon re-establishment of the connection 120, the client agent 128 and/or the first protocol service 112 can flush from each of their respective buffers the buffered data packets now known to be received by the other.

[0082] FIG. 5 depicts another illustrative embodiment of a system 400 that is capable of re-connecting the client 108 to a host service 116, as described above. In addition to the networks 104 and 104', the client 108, the first protocol service 112, and the host services 116, all of which are described above, the system 400 further includes an intermediary node 132, and a ticket authority 136. In one embodiment, the intermediary node 132 is a security gateway, such as, for example, a firewall and/or a router, through which messages between the client 108 and the first protocol service 112 must pass due to the configuration of the network 104. The ticket authority 136 can be, as illustrated, a stand-alone network component that is capable of communication and that has sufficient processor power and memory capacity to perform the operations described herein.

[0083] As shown in the illustrative embodiment of FIG. 5, the intermediary node 132 is configured to accept a connection 120a initiated by the client 108 and to establish a second connection 120b with the first protocol service 112. Together, the connection 120a and the second connection 120b constitute the connection 120, described above, over which the client 108 and the first protocol service 112 communicate using the first protocol.

[0084] The intermediary node 132, as shown, is also configured to communicate with the ticket authority 136. In one embodiment, the ticket authority 136 is configured to receive a request for a first re-connection ticket from the intermediate node 132 and to thereafter generate the first re-connection ticket. The first re-connection ticket can include, for example, a large random number.

[0085] In another embodiment, the ticket authority 136 is configured to generate a handle. The handle can be, for example, a random number that is associated with (*e.g.*, mapped to) the first re-connection ticket. In one embodiment, the handle is a smaller random number than the random number forming the first re-connection ticket. For example, the handle may be a 32-bit random number. The ticket authority 136 transmits the first re-connection ticket and the handle to the intermediary node 132, while keeping a copy of the first re-connection ticket and a copy of the handle. The copy of the first re-connection ticket can later be used by the ticket authority 136 to validate the first re-connection ticket originally transmitted to the client 108 when it is later presented to the ticket authority 136 during the process of re-connecting the client 108. In one embodiment, the ticket authority 136 also keeps an address for the first protocol service 112, which, as explained below, is associated with the first re-connection ticket and, upon validation of the first re-connection ticket, is transmitted to the intermediary node 132.

[0086] In one embodiment, the intermediary node 132 is further configured to use the handle transmitted to it by the ticket authority 136 to delete the copy of the first re-connection ticket kept at the ticket authority 136. In another embodiment, as described below, the ticket authority 136 is further configured to delete, during the process of re-connecting the client 108 to a host service 116, the first re-connection ticket and thereafter generate a replacement first re-

connection ticket. Additionally, in another embodiment, the first re-connection ticket is configured for automatic deletion after a pre-determined period of time.

[0087] In another embodiment, the first protocol service 112 is configured to generate a second re-connection ticket, which, as in the case of the first re-connection ticket, can include, for example, a large random number. The first protocol service 112 can also be configured to transmit the second re-connection ticket to the client 108, while keeping a copy of the second re-connection ticket and a session number. The copy of the second re-connection ticket can later be used by the first protocol service 112 to validate the second re-connection ticket originally transmitted to the client 108 when it is later presented to the first protocol service 112 during the process of re-connecting the client 108. In one embodiment, the first protocol service 112 transmits the second re-connection ticket to the client 108 via the intermediary node 132. In another embodiment, the first protocol service 112 transmits the second re-connection ticket to the client 108 directly. Moreover, as described in greater detail below, the first protocol service 112 can be further configured to delete, during the process of re-connecting the client 108 to a host service 116, the second re-connection ticket, and thereafter generate a replacement second re-connection ticket. Additionally, in another embodiment, the second re-connection ticket is configured for automatic deletion after a pre-determined period of time.

[0088] In one embodiment, the intermediary node 132 serves as an intermediary for the first and second re-connection tickets. The intermediary node 132 receives, for example, the first re-connection ticket generated by the ticket authority 136 and the second re-connection ticket generated by the first protocol service 112. The intermediary node 132 can then transmit the first re-connection ticket and the second re-connection ticket to the client 108. Moreover, during the process of re-connecting the client 108 to a host service 116, the intermediary node 132 can

accept the first re-connection ticket and the second re-connection ticket from the client 108 and thereafter transmit the first re-connection ticket to the ticket authority 136 and, if appropriate, the second re-connection ticket to the first protocol service 112.

[0089] The process of re-connecting the client 108 to a host service 116, and the use of the first and second re-connection tickets, will be further described by reference to the methods described below with reference to FIGS. 7-10.

[0090] Referring to FIG. 6A, another embodiment of a system 500 for network communications includes the networks 104 and 104', the client 108, the first protocol service 112, the host services 116, the intermediary node 132, and the ticket authority 136, as described above, and further depicts a first computing node 140 and a second computing node 144, both of which are used, in one embodiment, for initially connecting the client 108 to a host service 116. Moreover, in the illustrative embodiment of FIG. 6A, the client 108 further includes a web browser 148, such as, for example, the INTERNET EXPLORER program from Microsoft Corporation of Redmond, WA, to connect to the World Wide Web.

[0091] In one embodiment (not shown), the system 500 includes two or more intermediary nodes 132 and/or two or more first protocol services 112. The intermediary node 132, through which messages between the client 108 and the first protocol service 112 must pass, and/or the first protocol service 112 can, as explained below, each be chosen based on, for example, a load balancing equation.

[0092] Each of the first computing node 140 and the second computing node 144 can be any computing device that is capable of communication and that has sufficient processor power and memory capacity to perform the operations described herein. For example, in one embodiment,

the first computing node 140 is a web server, providing one or more websites. In another embodiment, the second computing node 144 provides an XML service.

[0093] In one embodiment, the client 108 and the network 104 form an external network 152, separated from the rest of the system 500 by a first firewall 156, depicted as a dashed line. The intermediary node 132 and the first computing node 140 can be located in a "demilitarized zone" 160 (*i.e.,* a network region placed between a company's private network and the public network), separated from the rest of the system 500 by the first firewall 156 and a second firewall 164, also depicted by a dashed line. Then, as shown, the network 104', the first protocol service 112, the host services 116a-116n, the ticket authority 136, and the second computing node 144, form an internal network 168, separated from the rest of the system 100 by the second firewall 164.

[0094] Alternatively, in another embodiment, and with reference to FIG. 6B, the system 500 further includes a third computing node 146 positioned, in the demilitarized zone 160, between the network 104 and the intermediary node 132. The third computing node 146 can be any computing device that is capable of networked communication and that has sufficient processor power and memory capacity to perform the operations described herein. As described below, the third computing node 146 is used, in some embodiments, during the process of initially connecting the client 108 to a host service 116 and/or during the process of re-connecting the client 108 to a host service 116. More specifically, as described below, where the system 500 includes two or more intermediary nodes 132, the third computing node 146 can, based on a load balancing equation for example, choose the intermediary node 132 through with communications between the client agent 128 of the client 108 and the first protocol service 112 must pass.

[0095] Moreover, referring to FIG. 6C, the intermediary node 132 of FIG. 6B can, in an alternative embodiment, be replaced by two or more levels "a"-"n" of intermediary nodes 132.

As illustrated, each level "a"-"n" can include two or more intermediary nodes 132a-132n. As described below, the client agent 128 of the client 108 can be routed through any combination of the intermediary nodes 132 based on, for example, load balancing equations. For example, as illustrated, the client agent 128 can be routed through the intermediary nodes 132 via connection 122. Other configurations of the system 500, as would be readily apparent to one skilled in the art, are also possible.

[0096] Referring again to FIG. 6A, in one embodiment, the web browser 148 communicates over the network 104 with the first computing node 140, which itself interfaces with the second computing node 144 and the ticket authority 136. More specifically, the first computing node 140 is configured with the address of the second computing node 144 and the ticket authority 136. In one embodiment, as explained further below, the first computing node 140 is configured to relay information between, and thereby prevent direct communication between, the web browser 148 of the client 108, the second computing node 144, and the ticket authority 136. By preventing such direct communication, the first computing node 140 adds an additional level of security to the system 500. The first computing node 140 can also be configured with the address of the intermediary node 132, or, alternatively, with the address of two or more intermediary nodes 132.

[0097] For its part, the second computing node 144 is configured to determine which of the application programs running on the host services 116 are available to a user of the client 108. In other words, the second computing node 144 is configured to determine which of the application programs the user is authorized to access. In one embodiment, after the user selects his desired application program, as described further below, the second computing node 144 is further configured to determine which of the host services 116 will be used to run the user's

desired application for purposes of load balancing. The second computing node 144 returns the address of that host service 116 to the first computing node 140. The second computing node 144 also returns the address of the first protocol service 112, which can also be selected from amongst a plurality of first protocol services 112 through the use of a load balancing equation, to the first computing node 140. In turn, the first computing node 140, transmits the address of the chosen first protocol service 112 and the chosen host service 116 to the ticket authority 136.

[0098] For its part, the ticket authority 136 generates connection tickets. In one embodiment, the ticket authority 136 transmits an initial connection ticket to the first computing node 140 for transmission to the client 108. In another embodiment, the ticket authority transmits a first re-connection ticket to the intermediary node 132.

[0099] The process of initially connecting the client 108 to the host service 116, and the roles of the ticket authority 136, the first computing node 140, the second computing node 144, and the third computing node 146 therefor, is explained below.

[0100] Referring now to FIG. 7, one embodiment of a method 600 for network communications, using the exemplary embodiment of FIGS. 6A-6C, is illustrated. At step 604, the client 108 initially connects to a plurality of host services 116 by employing, for example, the method 700 described below. After the client 108 is connected to the plurality of host services 116, the client 108 and the host services 116 communicate, through the first protocol service 112, and at step 608, via a plurality of secondary protocols encapsulated within the first protocol. In one embodiment, the first protocol service 112 encrypts, prior to the transmission of any first protocol packets, communications at the level of the first protocol 204, thereby securing the communications. In another embodiment, the first protocol service 112 compresses, prior to the

transmission of any first protocol packets, the communications at the level of the first protocol, thereby improving communication efficiency.

[0101] At step 612, the client agent 128 determines whether the connection 120 between the client agent 128 and the first protocol service 112 has failed. For example, the connection 120a between the client agent 128 and the intermediary node 132 may have failed, the connection 120b between the intermediary node 132 and the first protocol service 112 may have failed, or both the connection 120a and the connection 120b may have failed. If the client agent 128 determines that the connection 120 has not failed, the method 600 proceeds to step 620. If, on the other hand, the client agent 128 determines that the connection 120 has failed, the client 108 is, at step 616, provided with a reliable connection to the host services 116 and re-connected to the host services 116.

[0102] It is determined, at step 620, whether the client 108 wishes to cleanly terminate its connection 120 with the first protocol service 112 and, consequently, its connections 124a-124n with the host services 116. If not, communication between the client 108 and the first protocol service 112, via the plurality of secondary protocols encapsulated within the first protocol, continues at step 608. If so, then, at step 624, all connections 120a, 120b, and 124a-124n are broken and all re-connection tickets are deleted. In one embodiment, the intermediary node 132 uses a handle it receives from the ticket authority 136 to delete a copy of a first re-connection ticket kept at the ticket authority 136. In another embodiment, the first protocol service 112 deletes a copy of a second re-connection ticket kept at the first protocol service 112.

[0103] In a further embodiment, if for some reason a secondary protocol connection 124 fails, a copy of the second re-connection ticket associated therewith and kept at the first protocol service 112 is deleted by the first protocol service 112. In yet another embodiment, a first re-connection

ticket and/or a second re-connection ticket is automatically deleted after a pre-determined period of time following a failure in the connection 120, as at step 612, and/or following a clean termination of the connection 120, as at step 620.

[0104] Referring to FIGS. 8A-8C, one embodiment of a method 700 for initially connecting the client 108 to the host services 116 (for example at step 604 of FIG. 7), using the exemplary embodiment of FIG. 6A-6C, is illustrated.

[0105] At step 704, the client 108, using the browser 148, sends a request, such as, for example, an HTTP request, to the first computing node 140. The first computing node 140 returns a web page, such as, for example, an HTML form requesting authentication information (*e.g.*, a username and a password). A user of the client 108 enters his credentials and transmits the completed form to the first computing node 140.

[0106] The first computing node 140, at step 708, then informs the user of the client 108 of applications available for execution. In one embodiment, the first computing node 140 extracts the user's credentials from the login page and transmits them to the second computing node 144, together with a request for the second computing node 144 to enumerate the applications available to the user. Based on the user's credentials, the second computing node 144 returns a list of specific applications available to the user to the first computing node 140, which then forwards the list, in the form of a web page for example, to the user of the client 108.

[0107] At step 712, the user selects the desired application and a request for that application is sent to the first computing node 140. For example, in one embodiment, the user clicks on a desired application listed in the web page presented to him by the first computing node 140 and an HTTP request for that application is forwarded to the first computing node 140. The request is processed by the first computing node 140 and forwarded to the second computing node 144.

[0108] At step 716, the second computing node 144 determines the host service 116 on which the desired application will be executed. The second computing node 144 can make that determination based, for example, on a load balancing equation. In one embodiment, the second computing node 144 also determines a first protocol service 112 from amongst a plurality of first protocol services 112 that will be used to communicate with the host service 116 via a connection 124. Again, the second computing node 144 can make that determination based, for example, on a load balancing equation. The second computing node 144 returns the address of the chosen host service 116 and the chosen first protocol service 112 to the first computing node 140.

[0109] The client 108, at step 720, is then provided with an initial connection ticket and an address for the intermediary node 132 (which is either its actual address or its virtual address, as described below). In one embodiment, the first computing node 140 provides the address for the chosen host service 116 and the chosen first protocol service 112 to the ticket authority 136, together with a request for the initial connection ticket. The ticket authority 136 keeps the address of the chosen host service 116 and the chosen first protocol service 112, generates the initial connection ticket, and transmits the initial connection ticket to the first computing node 140, while keeping a copy for itself.

[0110] The first computing node 140, configured, in one embodiment, with the actual address of the intermediary node 132, then transmits the actual address of the intermediary node 132 and the initial connection ticket to the browser 148 of the client 108. The first computing node 140 can, for example, first create a file containing both the actual address of the intermediary node 132 and the initial connection ticket and then transmitting the file to the browser 148 of the client 108. Optionally, in another embodiment, the first computing node 140 is configured with the

actual address of two or more intermediary nodes 132. In such an embodiment, the first computing node 140 first determines the intermediary node 132 through which messages between the client 108 and the first protocol service 112 will have to pass. The first computing node 140 then transmits the actual address of that chosen intermediary node 132 and the initial connection ticket to the browser 148 of the client 108 using, for example, the file described above. In one embodiment, the first computing node 140 chooses the intermediary node 132 using a load balancing equation. The client agent 128 of the client 108 is then launched and uses the address of the intermediary node 132, to establish, at step 724, a first protocol connection 120a between the client agent 128 of the client 108 and the intermediary node 132.

[0111] Alternatively, in another embodiment, the first computing node 140 is configured with an actual address of the third computing node 146, which serves as a virtual address of an intermediary node 132. In such an embodiment, the first computing node 140 transmits, at step 720, the actual address of the third computing node 146 and the initial connection ticket to the browser 148 of the client 108 using, for example, the file described above. The client agent 128 of the client 108 is then launched and uses the actual address of the third computing node 146 to establish, at step 724, a first protocol connection between the client agent 128 of the client 108 and the third computing node 146. The third computing node 146 then determines the intermediary node 132 through which messages between the client 108 and the first protocol service 112 will have to pass. In one embodiment, the third computing node 146 chooses the intermediary node 132 using a load balancing equation. Having chosen the intermediary node 132, the third computing node 146 establishes a first protocol connection to the intermediary node 132. A first protocol connection 120a therefore exists, through the third computing node 146, between the client agent 128 of the client 108 and the intermediary node 132. The actual

address of the third computing node 146 is therefore mapped to the actual address of the intermediary node 132. To the client agent 128 of the client 108, the actual address of the third computing node 146 therefore serves as a virtual address of the intermediary node 132.

[0112] In one embodiment, where more than one level of intermediary nodes 132 exist, as described above, the first computing node 140 or the third computing node 146, respectively, only choose the intermediary node 132 to which the client agent 128 will connect at level "a." In such an embodiment, at each of the levels "a"-"n-1", the intermediary node 132 through which the client agent 128 is routed at that level thereafter determines, based on a load balancing equation for example, the intermediary node 132 to which it will connect at the next level. Alternatively, in other embodiments, the first computing node 140 or the third computing node 146, respectively, determine, for more than one or all of the levels "a"-"n", the intermediary nodes 132 through which the client agent 128 will be routed.

[0113] Having established the first protocol connection 120a between the client agent 128 of the client 108 and the intermediary node 132, for example the intermediate node 132 at level "n" (hereinafter referred to in method 700 as the intermediary node 132), the client agent 128 then transmits the initial connection ticket to the intermediary node 132.

[0114] It is then determined, at step 728, whether the initial connection ticket is valid. In one embodiment, the intermediary node 132 transmits the initial connection ticket to the ticket authority 136 for validation. In one embodiment, the ticket authority 136 determines the validity of the initial connection ticket by comparing it to the copy of the initial connection ticket it kept at step 720. If the ticket authority 136 determines the initial connection ticket to be valid, the ticket authority 136 transmits, at step 732, the address of the first protocol service 112 and the address of the chosen host service 116 to the intermediary node 132. The first protocol service

112 can also delete the initial connection ticket and the copy thereof. If, on the other hand, the ticket authority 136 determines the initial connection ticket to be invalid, the client 108 is, at step 730, refused connection to the first protocol service 112 and, consequently, connection to the host service 116.

[0115] Following step 732, the intermediary node 132 uses the address of the chosen first protocol service 112 to establish, at step 736, a first protocol connection 120b between the intermediary node 132 and the first protocol service 112. A first protocol connection 120 therefore now exists, through the intermediary node 132, between the client agent 128 of the client 108 and the first protocol service 112. The intermediary node 132 can also pass the address of the chosen host service 116 to the first protocol service 112.

[0116] In one embodiment, at step 740, the first protocol service 112 uses the address of the chosen host service 116 to establish a secondary protocol connection 124 between the first protocol service 112 and the chosen host service 116. For example, the chosen host service 116 is in fact the host service 116a and a secondary protocol connection 124a is established between the first protocol service 112 and the host service 116a.

[0117] In one embodiment, following step 740, the user chooses, at step 744, a second application to be executed and the second computing node 144 determines, at step 748, the host service 116 on which the second application is to be executed. For example, by calculating a load balancing equation, the second computing node 144 may choose the host service 116b to execute the second application program. The second computing node 144 then transmits the address of the chosen host service 116b to the first protocol service 112. In one embodiment, the second computing node 144 is in direct communication with the first protocol service 112 and directly transmits the address thereto. In another embodiment, the address of the chosen host

service 116b is indirectly transmitted to the first protocol service 112. For example, the address can be transmitted to the first protocol service 112 through any combination of the first computing node 140, the ticket authority 136, the intermediary node 132, and the first protocol service 112. Having received the address of the chosen host service 116b, the first protocol service 112 establishes, at step 752, a secondary protocol connection 124b between the first protocol service 112 and the chosen host service 116b.

[0118] The secondary protocols that can be used to communicate over the connections 124a and 124b include, but are not limited to, HTTP, FTP, Oscar, Telnet, ICA, and RDP. Moreover, in one embodiment, at least one of the secondary protocols, as described above, includes a plurality of virtual channels, each of which can include a plurality of protocol packets enabling functionality at the remote client 108. For example, in one embodiment, one host service 116a is a web server, communicating with the first protocol service 112 over the connection 124a using the HTTP protocol, and another host service 116b is an application server, communicating with the first protocol service 112 over the connection 124b using the ICA protocol. The host service 116b generates both protocol packets for transmitting graphical screen commands to the client 108, for causing the client 108 to display a graphical user interface, and protocol packets for transmitting printer commands to the client 108, for causing a document to be printed at the client 108.

[0119] Steps 744, 748, and 752 can be repeated any number of times. As such, any number of application programs can be executed on any number of host services 116a-116n, the outputs of which can be communicated to the first protocol service 112 over the connections 124a-124n using any number of secondary protocols.

[0120] Turning now to step 756, the first protocol service 112 can, as described above, encapsulate the plurality of secondary protocols within the first protocol. As such, the client 108 is connected to, and simultaneously communicates with, a plurality of host services 116.

[0121] In another embodiment, prior to performing steps 744, 748, and 752 to execute a new application program on a host service 116, such as, for example, the host service 116b, a user of the client 108 ends execution of another application program, such as, for example, an application program executing on host service 116a. In such a case, the first protocol service 112 disrupts the connection 124a between the first protocol service 112 and the host service 116a. The first protocol service 112 then establishes, by implementing steps 744, 748, and 752, the connection 124b between the first protocol service 112 and the host service 116b, without interrupting the connection 120 between the client 108 and the first protocol service 112.

[0122] In one embodiment, a first re-connection ticket is generated at step 760. For example, the intermediary node 132 requests a first re-connection ticket from the ticket authority 136. Upon receiving the request, the ticket authority 136 generates the first re-connection ticket, which is, for example, a large random number, and can also generate a handle, which is, for example, a smaller random number. The ticket authority 136 can then transmit, at step 764, the first re-connection ticket and the handle to the intermediary node 132, while keeping a copy of the first re-connection ticket and a copy of the handle. The ticket authority 136 continues to maintain the address of the first protocol service 112 that was transmitted to it by the first computing node 140 at step 720. The intermediary node 132 then transmits, at step 768, the first re-connection ticket to the client 108.

[0123] At step 772, a second re-connection ticket is then generated. In one embodiment, the first protocol service 112 generates the second re-connection ticket, which can be, for example, a

large random number. The first protocol service 112, at step 776, then transmits the second re-connection ticket, through the intermediary node 132, to the client 108. In doing so, the first protocol service 112 keeps a copy of the second re-connection ticket and a session number associated therewith for identifying the session to be re-connected following a disruption of the connection 120. In one embodiment, for example, the first protocol service 112 maintains, for a particular session number, a table listing the secondary protocol connections 124a-124n associated with that session number. Accordingly, following re-establishment of the first protocol connection 120 and validation of the second re-connection ticket at the first protocol service 112, as described below, the first protocol service 112 can identify the secondary protocol connections 124 to be encapsulated within the re-established first protocol connection 120 for communication to the client 108.

[0124] Alternatively, in another embodiment, and with reference again to FIG. 1A, the system 100 of the present invention does not include the intermediary node(s) 132, the ticket authority 136, nor the third computing node 146. In such an embodiment, rather than generating and transmitting, at steps 760 through 776, both the first and the second reconnection ticket, the system 100 and method 700 provide for only a single re-connection ticket. In one such embodiment, the first protocol service 112, for example, generates the single re-connection ticket, which can be, for example, a large random number. The first protocol service 112 then transmits the single re-connection ticket directly to the client 108 over the connection 120. In doing so, the first protocol service 112 keeps a copy of the single re-connection ticket and a session number associated therewith for identifying the session to be re-connected following a disruption of the connection 120.

[0125] Referring now to FIG. 9, one embodiment of a method 800 for providing a client 108 with a reliable connection to one or more host services 116 and for re-connecting the client 108 to the host services 116 (for example at step 616 of FIG. 7), using the exemplary embodiment of FIGS. 6A-6C, is illustrated. In particular, at step 804, the secondary protocol connection 124 between the first protocol service 112 and each of the one or more host services 116 is maintained. Moreover, at step 808, a queue of data packets most recently transmitted between the client agent 128 of the client 108 and the first protocol service 112, via the connection 120 that was determined to have broken, for example, at step 616 of FIG. 7, is maintained. In one embodiment, the data packets are queued and maintained both before and upon failure of the connection 120. The queued data packets can be maintained, for example, in a buffer by the client agent 128. Alternatively, the first protocol service 112 can maintain in a buffer the queued data packets. In yet another embodiment, both the client agent 128 and the first protocol service 112 maintain the queued data packets in a buffer.

[0126] At step 812, a new first protocol connection 120 is established between the client agent 128 of the client 108 and the first protocol service 112 and linked to the maintained secondary protocol connection 124 between the first protocol service 112 and each of the one or more host services 116, thereby re-connecting the client 108 to the host services 116. After the client 108 is re-connected, the queued data packets maintained at step 808 can be transmitted, at step 816, via the newly established first protocol connection 120. As such, the communication session between the host services 116 and the client 108, through the first protocol service 112, is persistent and proceeds without any loss of data.

[0127] Referring now to FIGS. 10A-10B, one embodiment of a method 900 for re-connecting the client 108 to the one or more host services 116 (for example at step 812 of FIG. 9), using the exemplary embodiment of FIGS. 6A-6C, is illustrated.

[0128] At step 904, any remaining connections between the client 108 and the first protocol service 112 are broken. For example, where the connection 120a has failed, but the connection 120b has not, the connection 120b is broken. Alternatively, where the connection 120b has failed, but the connection 120a has not, the connection 120a is broken.

[0129] In one embodiment, using the actual address of the intermediary node 132 provided to the client 108, for example at step 720 of FIG. 8, the client agent 128 of the client 108 then re-establishes, at step 908, the first protocol connection 120a between the client agent 128 and the intermediary node 132. Alternatively, in another embodiment, using the actual address of the third computing node 146 provided to the client 108, for example at step 720 of FIG. 8, the client agent 128 of the client 108 then re-establishes, at step 908, a first protocol connection between the client agent 128 and the third computing node 146. The third computing node 146 then determines the intermediary node 132 through which messages between the client 108 and the first protocol service 112 will have to pass. In one embodiment, the third computing node 146 chooses the intermediary node 132 using a load balancing equation. The intermediary node 132 chosen by the third computing node 146 in re-connecting the client 108 to the one or more host services 116 can be different from that chosen, for example at step 720 of FIG. 8, to initially connect the client 108 to the one or more host services 116. Having chosen the intermediary node 132, the third computing node 146 re-establishes a first protocol connection to the intermediary node 132. A first protocol connection 120a is therefore re-established, through the

third computing node 146, between the client agent 128 of the client 108 and the intermediary node 132.

[0130] In one embodiment, where more than one level of intermediary nodes 132 exist, the intermediary node 132 through which the client agent 128 is routed at each of the levels "a"-"n-1" thereafter determines, based on a load balancing equation for example, the intermediary node 132 to which it will connect at the next level. Alternatively, in another embodiment, the third computing node 146 determines, for more than one or all of the levels "a"-"n", the intermediary nodes 132 through which the client agent 128 will be routed.

[0131] Having re-established the first protocol connection 120a between the client agent 128 of the client 108 and the intermediary node 132, for example the intermediate node 132 at level "n" (hereinafter referred to in method 900 as the intermediary node 132), the client agent 128 then transmits, at step 912, the first re-connection ticket and the second re-connection ticket to the intermediary node 132.

[0132] It is then determined, at step 916, whether the first re-connection ticket is valid. In one embodiment, the validity of the first re-connection ticket is determined by using the ticket authority 136. For example, the intermediary node 132 transmits the first re-connection ticket to the ticket authority 136. In one embodiment, the ticket authority 136 determines the validity of the first re-connection ticket by comparing it to a previously kept copy of the first re-connection ticket. If the ticket authority 136 determines the first re-connection ticket to be valid, the ticket authority 136 transmits, at step 920, the address of the first protocol service 112 to the intermediary node 132. Otherwise, if the ticket authority 136 determines the first re-connection ticket to be invalid, the client 108 is, at step 924, refused re-connection to the first protocol service 112 and, consequently, re-connection to the host services 116.

[0133] At step 928, the first re-connection ticket is deleted by, for example, the ticket authority 136 and a replacement first re-connection ticket is generated by, for example, the ticket authority 136. Moreover, a replacement handle can be generated by, for example, the ticket authority 136. In some such embodiments, the ticket authority 136 transmits the replacement first re-connection ticket and the replacement handle to the intermediary node 132. Moreover, in some such embodiments, the ticket authority 136 keeps a copy of the replacement first re-connection ticket. In some embodiments, the ticket authority 136 waits for the client 108 to acknowledge that it has received the replacement first re-connection ticket before it proceeds to delete the first re-connection ticket.

[0134] After the first re-connection ticket is validated, the intermediary node 132, using the address of the first protocol service 112, re-establishes, at step 932, the first protocol connection 120b between the intermediary node 132 and the first protocol service 112. Having re-established the first protocol connection 120b between the intermediary node 132 and the first protocol service 112, it is then determined, at step 936, whether the second re-connection ticket is valid. In one embodiment, the validity of the second re-connection ticket is determined by using the first protocol service 112. For example, the intermediary node 132 transmits the second re-connection ticket to the first protocol service 112. In one embodiment, the first protocol service 112 determines the validity of the second re-connection ticket by comparing it to a previously kept copy of the second re-connection ticket. If the first protocol service 112 determines the second re-connection ticket to be valid, the re-established first protocol connection 120b between the first intermediary node 132 and the first protocol service 112 is linked, at step 940, to the maintained secondary protocol connection 124 between the first protocol service 112 and each of the one or more host services 116. Otherwise, if the first

protocol service 112 determines the second re-connection ticket to be invalid, the re-established first protocol connection 120b is not linked to the one or more maintained secondary protocol connections 124 and the client 108 is, at step 944, refused re-connection to the one or more host services 116.

[0135] At step 948, the second re-connection ticket is deleted by, for example, the first protocol service 112 and a replacement second re-connection ticket is generated by, for example, the first protocol service 112 for transmission to the client 108. In such an embodiment, the first protocol service 112 keeps a copy of the replacement second re-connection ticket. In some embodiments, the first protocol service 112 waits for the client 108 to acknowledge that it has received the replacement second re-connection ticket before it proceeds to delete the second re-connection ticket.

[0136] At step 952, the replacement first re-connection ticket and the replacement second re-connection ticket are transmitted to the client. For example, the ticket authority 136 can transmit, through the intermediary node 132, the replacement first re-connection ticket to the client 108. Moreover, in one embodiment, the first protocol service 112 transmits, through the intermediary node 132, the replacement second re-connection ticket to the client 108.

[0137] Alternatively, in other embodiments, as discussed above, the system 100 and methods of the invention provide for only a single re-connection ticket. As such, rather than using both first and second re-connection tickets, the method 900 of the invention uses only the aforementioned single re-connection ticket. In one such embodiment, the client agent 128 of the client 108 is also provided with the address of the first protocol service 112. To re-connect to the host services 116, the client agent 128 transmits the single re-connection ticket directly to the first protocol service 112. The first protocol service 112 then determines whether the single re-

connection ticket is valid. In one embodiment, the first protocol service 112 determines the validity of the single re-connection ticket by comparing it to a previously kept copy of the single re-connection ticket. If the first protocol service 112 determines the single re-connection ticket to be valid, the re-established first protocol connection 120 between the client 108 and the first protocol service 112 is linked to the maintained secondary protocol connection 124 between the first protocol service 112 and each of the one or more host services 116. Otherwise, if the first protocol service 112 determines the single re-connection ticket to be invalid, the re-established first protocol connection 120 is not linked to the one or more maintained secondary protocol connections 124 and the client 108 is refused re-connection to the one or more host services 116.

[0138] After the single re-connection ticket is validated, the single re-connection ticket is deleted by, for example, the first protocol service 112 and a replacement single re-connection ticket is generated by, for example, the first protocol service 112 for transmission to the client 108. In transmitting the replacement single re-connection ticket to the client 108, the first protocol service 112 keeps a copy of the replacement single re-connection ticket. In some embodiments, the first protocol service 112 waits for the client 108 to acknowledge that it has received the replacement single re-connection ticket before it proceeds to delete the single re-connection ticket.

[0139] In yet another embodiment, like the first and second re-connection tickets, the single re-connection ticket is configured for automatic deletion after a pre-determined period of time following a failure in the connection 120, as at step 612, and/or following a clean termination of the connection 120, as at step 620.

[0140] Variations, modifications, and other implementations of what is described herein will occur to those of ordinary skill in the art without departing from the spirit and the scope of the invention. The invention is not to be defined only by the preceding illustrative description.

[0141] What is claimed is: